A New Nonlinear Equations Test Problem

J.E. Dennis, Jr.

David M. Gay**
and
Phuong Ahn Vu***

Technical Report 83-16, December 1985.

Mathematical Sciences Department, Rice University, Houston, Texas 77251-1892. Research sponsored by NSF MCS81-16779, DOE DE-AS05-82ER13016, ARO DAAG-29-83-K-0035.

[1]AT&T Bell Laboratories, Murray Hill, N.J. 07974. Research sponsored by NSF MCS81-16779.

[2]*J.S. Nolan Associates, Houston, Texas.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
DEC 1985		**00-00-1985 to 00-00-1985**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A New Nonlinear Equations Test Problem	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Computational and Applied Mathematics Department ,Rice University,6100 Main Street MS 134,Houston,TX,77005-1892**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**12**	

A New Nonlinear Equations Test Problem

*J.E. Dennis, Jr.**

*David M. Gay***

*Phuong Ahn Vu****

ABSTRACT

This report presents a set of test problems for nonlinear equations and nonlinear least-squares algorithms. These problems, sent to us by C.V. Nelson of the Maine Medical Center, come from a dipole model of the heart. They are 6×6 or 8×8, easy to code, cheap to evaluate, and not easy to solve. In support of the latter contention, we present test results from MINPACK and NL2SOL.

1. Introduction

Consider the following two problems:

NLEQ: Given $F:\mathbf{R}^n \to \mathbf{R}^n$, solve $F(x) = 0$.

NLLS: Given $F:\mathbf{R}^n \to \mathbf{R}^m$, minimize $\phi(x) = \frac{1}{2} F(x)^T F(x)$.

Most algorithms for the solution of these two problems are based on the assumption that F can be adequately modeled by an affine function in some neighborhood of a point of interest, whether that point is close to or far away from the solution to the problem. The purpose of this note is first to add an interesting test function to the current list of test problems for nonlinear equations and nonlinear least squares, and second to use that test function to give some indications that affine modeling by the first two terms of the Taylor series is not necessarily the best strategy for Newton-type methods far from the solution to the problem. This claim will be supported by computational results from five codes:

- HYBRD from MINPACK — intended for NLEQ — uses a secant affine model for F;
- f-d HYBRD from MINPACK — intended for NLEQ — uses a Newton affine model for F;
- LMDIF from MINPACK — intended for NLLS — uses a Newton affine model for F;
- NL2SNO from NL2SOL — intended for NLLS — uses a mixed quadratic model for ϕ;
- DN2F — intended for NLLS — a slight modification of NL2SOL.

In the next section, we will give a brief description of the test problems mentioned above. Section 3 will discuss the way the five codes handle their corresponding problems and Section 4 gives the computational results together with some conclusions. A listing of the Fortran subroutine

*Mathematical Sciences Department, Rice University, Houston, Texas 77251. Research sponsored by NSF MCS81-16779, DOE DE-AS05-82ER13016, ARO DAAG-29-83-K-0035. This work was supported in part by the International Business Machine Corporation, Palo Alto Scientific Center, Palo Alto, CA.

**AT&T Bell Laboratories, Murray Hill, N.J. 07974. Research sponsored by NSF MCS81-16779.

***J.S. Nolan Associates, Houston, Texas. This work was supported in part by the International Business Machine Corporation, Palo Alto Scientific Center, Palo Alto, CA.

used to evaluate the test function appears in the appendix, as do five sets of experimental data and corresponding solutions.

2. The Test Problem

We give here a brief description of a problem communicated to us by C.V. Nelson (1980). The interested reader can find a detailed description of the problem together with the derivation of the equations in Nelson and Hodgkin (1981). The problem comes from experiments using two artificial dipoles in a circular disk containing electrolyte to determine the resultant dipole moment of the human heart. Potentials from electrodes around the boundary of the disk are measured and the Gabor-Nelson (1954) equations are used to solve for the magnitude, directions, and locations of the two independent dipoles in the disks. The problem reduces to that of solving the following 8×8 system of nonlinear equations in the unknowns a, b, c, d, t, u, v, w:

$$a + b = \Sigma M_x$$

$$c + d = \Sigma M_y$$

$$ta + ub - vc - wd = \Sigma A$$

$$va + wb + tc + ud = \Sigma B$$

$$a(t^2 - v^2) - 2ctv + b(u^2 - w^2) - 2duw = \Sigma C$$

$$c(t^2 - v^2) + 2atv + d(u^2 - w^2) + 2buw = \Sigma D$$

$$at(t^2 - 3v^2) + cv(v^2 - 3t^2) + bu(u^2 - 3w^2) + dw(w^2 - 3u^2) = \Sigma E$$

$$ct(t^2 - 3v^2) - av(v^2 - 3t^2) + du(u^2 - 3w^2) - bw(w^2 - 3u^2) = \Sigma F,$$

where the right-hand sides of the equations in each experiment were evaluated from measured potentials. Note that since the first two equations are linear in the variables, we can eliminate two of the first four variables to obtain a 6×6 system which might be easier to solve. Below, we will assume that the equations have been rewritten to have right hand side equal to zero and we will refer to the 8×8 system as the *full problem* and to the 6×6 system as the *reduced problem*.

This appears to be a good test problem for NLEQ and NLLS algorithms because it is easy to code, cheap to evaluate, and hard to solve, judging from our experience with the five codes listed above. In all of the problems, we did a LINPACK-type condition estimate of the Jacobian matrix $F'(x)$ at the solution and it was never greater that 10^5.

3. The Codes Tested

In this section we will give a brief description of the five codes we tested. We used the finite-difference version of each. All five codes use model-trust region algorithms; for detailed explanations of such algorithms, see Chapters 6, 8, and 10 of Dennis and Schnabel (1983).

Only two of the five codes tested, HYBRD and f-d HYBRD, use the $n\times n$ structure of the problem. They both use a modification of Powell's (1970) dogleg algorithm as a global strategy, and the local strategy is based on an affine model of $F(x)$. In f-d HYBRD this is done using forward differences. In HYBRD, the model is constructed at each iterate using the Broyden (1965) secant update to approximate the Jacobian $F'(x)$. Broyden's scheme provides a model that matches the two most current F-values rather than the current F and F' values. This indicates that perhaps the Broyden scheme is better able to remember the general shape of the function over several iterations far from the solution, and conversely, less able to forget outdated information as the iteration proceeds. The initial model Jacobian is provided by finite differences and if the updated affine model seems to be too inaccurate for reasonable progress at any iteration, then the Jacobian

approximation is refreshed by finite differences. When the Broyden update is used, the linear algebra to compute an iterate only requires $O(n^2)$ flops.

The other three programs consider the $n \times n$ systems as nonlinear least-squares problems, i.e., instead of solving $F(x) = 0$, they solve min $\frac{1}{2}F(x)^T F(x)$. By posing the problems in this way, we are able to use the LMDIF code from MINPACK and the NL2SNO code from the NL2SOL package to obtain a solution. As a result of these experiments, we discovered that NL2SOL was computing a lower bound for the Marquardt parameter that was smaller than intended. DN2F is a variant of NL2SOL that appears in PORT 3, the third edition of the subroutine library described by Fox, Hall, and Schryer (1978). It is similar to NL2SOL, except that the bounds on the Marquardt parameter are computed as in LMDIF, the trust radius is sometimes not changed at the beginning of an iteration (specifically, the trust radius is not allowed to decrease if the last step was a "good" one), and scaling (i.e., the choice of D^k in (3.1)) is based on the infinity-norm of the Jacobian matrix columns rather than their Euclidean norm. All three of these codes require $O(n^3)$ flops to do the linear algebra at each iteration. LMDIF uses a modification of the Levenberg-Marquardt algorithm, i.e., given a current estimate x_k to the solution, it determines a search direction s_k and subsequent iterate x_{k+1} by

$$(J_k^T J_k + \mu_k D_k) s_k = -J_k^T F(x_k) \tag{3.1}$$

$$x_{k+1} = x_k + s_k$$

where $J_k = F'(x_k)$, D_k is a positive diagonal matrix, and μ_k is a nonnegative scalar that is adaptively chosen.

Thus LMDIF builds a local quadratic model of ϕ by matching the current functional and gradient values and using $J(x_k)^T J(x_k)$ as a Hessian approximation. We call this the Gauss-Newton model of ϕ, and it is the quadratic model obtained by building a Newton affine model of $F(x)$ at x_k, i.e., a model that matches F and F' at x_k, and then taking the sum of squares of the affine model. This Gauss-Newton model of ϕ only coincides with the full Newton quadratic model of ϕ if F is truly affine or zero in each residual. However, the problems being considered here have zero residuals at the solution, so the Gauss-Newton method will have the same quadratic convergence as Newton's method near the solution.

The Hessian approximation used by NL2SNO and DN2F includes a cheap variable-metric secant approximation to the part of the Hessian of ϕ that the Gauss-Newton model neglects. Thus, it is a compromise between the Gauss-Newton use of information only at x_k and the secant method's memory of past F-values. The algorithm decides adaptively at each iteration whether to use this Hessian augmentation. The decision is based on quadratic model information from a trust-region implementation. For details the reader should consult Dennis, Gay, and Welsch (1981) or Dennis and Schnabel (1983).

Both the NLLS algorithms use basically the same global strategy. It is usually viewed as obtaining the next iterate by minimizing a local quadratic model of ϕ in some region about the current iterate where the model can be reasonably trusted to adequately model $\phi(x)$. The only difference is that LMDIF never has to worry about encountering negative curvature in the model and so the computation of each iterate is a bit simpler. Details for LMDIF can be found in Moré (1978) or in Chapter 6 of Dennis and Schnabel (1983). Details for NL2SNO can be found in Gay (1981).

All three implementations compute an approximate Jacobian matrix by forward differences using a step size in the ith coordinate direction of $|x_i|(machep)^{1/2}$, where *machep* is the machine epsilon or unit rounding error of the arithmetic. Also, all three codes have the capability of either letting the user choose a fixed diagonal scaling-matrix for the independent variable, or determining it internally and updating it at each iteration.

4. Computational Results

Everything is coded in Fortran and run on a VAX®11/750 under a UNIX® operating system using DOUBLE PRECISION arithmetic (for which we take *machep* = 2^{-55}). Both the full and reduced problems are run with and without scaling of the independent variable (i.e., with D_k in (3.1) chosen dynamically by the algorithm and with $D_k \equiv I$). Also, we execute all five algorithms using as initial guess x_0, $10x_0$, and $100x_0$, where x_0 is the experimental data provided by Nelson and given in the appendix. Since there are five sets of data, this gives us a total of 60 test cases. We always found the same single solution for each experiment, independent of initial guess, scaling option, and whether or not we solved the full or reduced system. In comparing off-the-shelf codes, there is always the problem of differing stopping conditions. We used the same tolerances for the same tests whenever possible. The convergence tests used by the NLLS codes are quite similar, but our version of HYBRD uses only a test on the relative change in the dependent variable. The other codes use this test also; the tolerance was set to $machep^{1/2}$ for all the runs.

To obtain the results reported for HYBRD, we had to comment out its tests for lack of progress. Without this change, HYBRD failed on about a third of the test problems. We did not have to make this change, however, to f-d HYBRD, which we had obtained by modifying HYBRD to force a new finite-difference calculation of the Jacobian after every successful step.

Output from NL2SNO and DN2F tells us that the algorithms tended to use the augmented Gauss-Newton model far away from the solution and the traditional Gauss-Newton model near the solution. This seems to suggest that the augmentation of the Gauss-Newton Hessian not only plays an important role in large-residual problems, but also gives a better approximation to the least-squares Hessian in the small-residual case when we are far from the solution. Perhaps our earlier discussion of the possible advantages of memory in the secant methods is relevant here.

On the other hand, the occasional efficiency of DN2F suggests that sometimes it would be worthwhile in the nonlinear equations problem to find an analog to the NL2SOL secant augmentation, sizing, and model switching so that memory of past points can be suppressed as the solution is approached, while providing a better model than the affine approximation for the function in the early iterations. A sophisticated new approach to this problem based on tensor updating is suggested in Schnabel and Frank (1984).

VAX is a trademark of Digital Equipment Corporation.
UNIX is a trademark of AT&T Bell Laboratories.

Table 1 - Total number of residual calculations with scaling.

		HYBRD		LMDIF		NL2SNO(*)		DN2F(*)		f-d HYBRD	
		Full Prob.	Red. Prob.	Full Prob.	Red. Prob.	Full Prob.	Red. Prob.	Full Prob.	Red. Prob.	Full Prob.	Red. Prob.
I	x_0	123	137	327	217	286	198	309	340	225	242
	$10x_0$	188	127	426	211	381	170	346	186	337	192
	$100x_0$	170	104	361	219	391	228	327	223	325	196
II	x_0	17	15	37	29	45	35	45	35	37	29
	$10x_0$	1292	61	109	92	1889	658	108	91	109	92
	$100x_0$	1902	438	630	356	4302	1407	717	313	235	177
III	x_0	311	273	1348	696	1454	641	1082	705	295	448
	$10x_0$	2452	863	3525	963	4904	1108	4126	610	428	419
	$100x_0$	272	308	4278	1916	5338	3684	5035	7117	451	577
IV	x_0	1015	666	1208	626	1659	569	1303	551	211	257
	$10x_0$	795	661	755	3063	2723	2760	3827	671	175	486
	$100x_0$	310	192	1951	1183	5199	2834	1229	809	302	305
V	x_0	385	419	1082	508	1747	378	1076	415	545	1289
	$10x_0$	1135	138	978	400	3730	630	1082	459	178	228
	$100x_0$	232	143	1593	1004	5072	3197	1686	636	226	291

I: Exp. 791129 III: Exp. 0121a V: Exp. 0121c
II: Exp. 791226 IV: Exp. 0121b

(*) Both NL2SN0 and DN2F were run with the LMDIF default initial step bound of $100 * \|D_0 * x_0\|$. Also V(RDFCMX), the maximum factor by which the trust region radius may be increased at one time is changed from the default value of 4 to be the same as the LMDIF default value of 2.

Table 2 - Total number of residual calculations without scaling.

		HYBRD		LMDIF		NL2SNO(*)		DN2F(*)		f-d HYBRD	
		Full Prob.	Red. Prob.	Full Prob.	Red. Prob.	Full Prob.	Red. Prob.	Full Prob.	Red. Prob.	Full Prob.	Red. Prob.
I	x_0	180	126	700	443	552	344	439	255	372	262
	$10x_0$	195	115	340	205	1269	183	274	172	289	207
	$100x_0$	144	109	379	219	660	172	309	222	327	196
II	x_0	17	15	37	29	45	35	45	35	37	29
	$10x_0$	336	423	109	138	1115	356	108	91	109	85
	$100x_0$	3806	401	2166	683	9027	1594	664	335	523	145
III	x_0	784	379	1479	724	1904	500	1339	640	430	434
	$10x_0$	294	590	2255	2527	3924	2039	4144	2288	397	727
	$100x_0$	716	277	966	1476	13736	1990	4818	7116	524	149
IV	x_0	399	426	1414	728	1228	506	1284	419	461	542
	$10x_0$	278	1191	880	5004	3764	3087	3263	3186	204	403
	$100x_0$	870	311	3107	1942	19373	2384	1285	815	195	143
V	x_0	303	172	1326	529	2153	905	1264	445	163	151
	$10x_0$	1006	375	939	1041	4596	1623	840	834	158	2083
	$100x_0$	1430	306	1010	2235	23801	2749	2320	399	236	144

I: Exp. 791129 III: Exp. 0121a V: Exp. 0121c
II: Exp. 791226 IV: Exp. 0121b

(*) Both NL2SN0 and DN2F were run with the LMDIF default initial step bound of $100 * \|x_0\|$. Also V(RDFCMX), the maximum factor by which the trust region radius may be increased at one time is changed from the default value of 4 to be the same as the LMDIF default value of 2.

5. References

C.G. Broyden (1965), "A class of methods for solving nonlinear simultaneous equations", *Math. Comp. 19*, pp. 577–593.

J.E. Dennis Jr., D.M. Gay, and R.E. Welsch (1981a), "An adaptive nonlinear least-squares algorithm", *ACM Trans. Math. Software 7*, pp. 348–368.

J.E. Dennis Jr., D.M. Gay, and R.E. Welsch (1981b), "Algorithm 573 NL2SOL — An adaptive nonlinear least-squares algorithm [E4]", *ACM Trans. Math. Software 7*, pp. 369–383.

J.E. Dennis Jr. and R.B. Schnabel (1983), *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ.

P.A. Fox, A.D. Hall and N.L. Schryer (1978), "The PORT Mathematical Subroutine Library", ACM Trans. Math. Software 4, pp. 104–126.

D. Gabor and C.V. Nelson (1954), "Determination of the resultant dipole of the heart from measurements on the body surface", *J.Appl.Physics, 25*, pp. 413–416.

D.M. Gay (1981), "Computing optimal locally constrained steps", *SIAM J. Sci. Stat. Comp. 2*, pp. 186–197.

J.J. Moré (1978), "The Levenberg-Marquardt algorithm: implementation and theory", in *Numerical Analysis*, G.A. Watson, Ed., Lecture Notes in Math. 630, Springer Verlag, Berlin, pp. 105–116.

J.J. Moré, B.S. Garbow, and K.E. Hillstrom (1980), "User guide for MINPACK–1", Argonne National Labs Report ANL–80–74.

C.V. Nelson (1980), private communication.

C.V. Nelson and B.C. Hodgkin (1981), "Determination of magnitudes, directions and locations of two independent dipoles in a circular conducting region from boundary potential measurements", *IEEE Trans Biomed Eng, BME–28*, pp. 817–823.

M.J.D. Powell (1970), "A hybrid method for nonlinear equations", in *Numerical Methods for Nonlinear Algebraic Equations*, P.Rabinowitz, Ed., Gordon and Breach, London, pp. 87–114.

R.B. Schnabel and P.D. Frank (1984), "Tensor methods for nonlinear equations", *SIAM J. Numer. Anal. 21*, pp. 815–843.

6. Appendix

Let $\Sigma = (\sum M_x, \sum M_y, \sum A, \sum B, \sum C, \sum D, \sum E, \sum F)^T$.

Exp. 791129

$$\Sigma = (.485, -.0019, -.0581, .015, .105, .0406, .167, -.399)^T$$
$$X_0 = (.299, .186, -.0273, .0254, -.474, .474, -.0892, .0892)^T$$

$$X^* \approx (-6.321349025e-3,\ 4.913213490e-1,\ -1.998156408e-3,\ 9.815640840e-5$$
$$1.226569755e-1,\ -1.003153205e-1,\ -4.023517593e+0,\ -2.071785527e-2)^T$$

Exp. 791226

$$\Sigma = (-.69, -.044, -1.57, -1.31, -2.65, 2.0, -12.6, 9.48)^T$$
$$X_0 = (-.3, -.39, .3, -.344, -1.2, 2.69, 1.59, -1.5)^T$$

$$X^* \approx (-3.116266056e-1,\ -3.783733944e-1,\ 3.282442301e-1,\ -3.722442301e-1$$
$$-1.282227094e+0,\ 2.494300312e+0,\ 1.554865879e+0,\ -1.384637843e+0)^T$$

Exp. 0121a

$$\Sigma = (-.816, -.017, -1.826, -.754, -4.839, -3.259, -14.023, 15.467)^T$$
$$X_0 = (-.041, -.775, .03, -.047, -2.565, 2.565, -.754, .754)^T$$

$$X^* \approx (3.099869097e-3,\ -8.190998691e-1,\ -2.239405352e-4,\ -1.677605946e-2$$
$$2.681514498e+0,\ 2.250215931e+0,\ -2.024170463e+1,\ 7.970982952e-1)^T$$

Exp. 0121b

$$\Sigma = (-.809, -.021, -2.04, -.614, -6.903, -2.934, -26.328, 18.639)^T$$
$$X_0 = (-.056, -.753, .026, -.047, -2.991, 2.991, -.568, .568)^T$$

$$X^* \approx (9.034542990e-3,\ -8.180345430e-1,\ -4.450738446e-4,\ -2.055492616e-2$$
$$2.773429036e+0,\ 2.529477259e+0,\ -1.480097186e+1,\ 5.220468844e-1)^T$$

Exp. 0121c

$$\Sigma = (-.807, -.021, -2.379, -.364, -10.541, -1.961, -51.551, 21.053)^T$$
$$X_0 = (-.074, -.733, .013, -.034, -3.632, 3.632, -.289, .289)^T$$

$$X^* \approx (5.140417418e-2,\ -8.584041742e-1,\ 1.047333626e-3,\ -2.204733363e-2$$
$$2.861205288e+0,\ 2.949155438e+0,\ -8.304243489e+0,\ -1.454992413e-1)^T$$

```
      SUBROUTINE FCN(N, X, FVEC, IFLAG)
C
C     This is an example of a subroutine to evaluate the "heart model"
C     function to use with HYBRD of MINPACK.
C
C    IPICK = 1 ---> full problem.
C    IPICK = 2 ---> reduced problem.
C
      INTEGER N, IFLAG
      DOUBLE PRECISION X(N),FVEC(N)
C
      INTEGER IPICK
      DOUBLE PRECISION SIGMAX, SIGMAY, SIGMAA, SIGMAB, SIGMAC, SIGMAD,
     1                 SIGMAE, SIGMAF
      COMMON  /SIGMA/  SIGMAX, SIGMAY, SIGMAA, SIGMAB, SIGMAC, SIGMAD,
     1                 SIGMAE, SIGMAF, IPICK
C
      DOUBLE PRECISION AT, ATV, AV, B, BU, BUW, BW, CT, CTV, CV, D, DU,
     1            DUW, DW, T2M3V2, T2MV2, U2M3W2, U2MW2, V2M3T2, W2M3U2
C
      IF (IPICK .NE. 1) GO TO 10
C
      T2MV2  = X(5)**2 - X(7)**2
      U2MW2  = X(6)**2 - X(8)**2
      T2M3V2 = X(5)**2 - 3.0D0 * X(7)**2
      V2M3T2 = X(7)**2 - 3.0D0 * X(5)**2
      U2M3W2 = X(6)**2 - 3.0D0 * X(8)**2
      W2M3U2 = X(8)**2 - 3.0D0 * X(6)**2
      CTV = X(3) * X(5) * X(7)
      DUW = X(4) * X(6) * X(8)
      ATV = X(1) * X(5) * X(7)
      BUW = X(2) * X(6) * X(8)
      AT  = X(1) * X(5)
      BU  = X(2) * X(6)
      CV  = X(3) * X(7)
      DW  = X(4) * X(8)
      CT  = X(3) * X(5)
      AV  = X(1) * X(7)
      DU  = X(4) * X(6)
      BW  = X(2) * X(8)
C
      FVEC(1) = X(1) + X(2) - SIGMAX
      FVEC(2) = X(3) + X(4) - SIGMAY
      FVEC(3) = AT + BU - CV - DW - SIGMAA
      FVEC(4) = AV + BW + CT + DU - SIGMAB
      FVEC(5) = X(1)*T2MV2 - 2.0D0*CTV + X(2)*U2MW2 - 2.0D0*DUW - SIGMAC
      FVEC(6) = X(3)*T2MV2 + 2.0D0*ATV + X(4)*U2MW2 + 2.0D0*BUW - SIGMAD
      FVEC(7) = AT*T2M3V2 + CV*V2M3T2 + BU*U2M3W2 + DW*W2M3U2 - SIGMAE
      FVEC(8) = CT*T2M3V2 - AV*V2M3T2 + DU*U2M3W2 - BW*W2M3U2 - SIGMAF
      GO TO 999
C
 10   T2MV2  = X(3)**2 - X(5)**2
      U2MW2  = X(4)**2 - X(6)**2
      T2M3V2 = X(3)**2 - 3.0D0 * X(5)**2
```

```
      V2M3T2 = X(5)**2 - 3.0D0 * X(3)**2
      U2M3W2 = X(4)**2 - 3.0D0 * X(6)**2
      W2M3U2 = X(6)**2 - 3.0D0 * X(4)**2
      B = SIGMAX - X(1)
      D = SIGMAY - X(2)
      CTV = X(2) * X(3) * X(5)
      DUW = D * X(4) * X(6)
      ATV = X(1) * X(3) * X(5)
      BUW = B * X(4) * X(6)
      AT  = X(1) * X(3)
      BU  = B * X(4)
      CV  = X(2) * X(5)
      DW  = D * X(6)
      CT  = X(2) * X(3)
      AV  = X(1) * X(5)
      DU  = D * X(4)
      BW  = B * X(6)
C
      FVEC(1) = AT + BU - CV - DW - SIGMAA
      FVEC(2) = AV + BW + CT + DU - SIGMAB
      FVEC(3) = X(1)*T2MV2 - 2.0D0*CTV + B*U2MW2 - 2.0D0*DUW - SIGMAC
      FVEC(4) = X(2)*T2MV2 + 2.0D0*ATV + D*U2MW2 + 2.0D0*BUW - SIGMAD
      FVEC(5) = AT*T2M3V2 + CV*V2M3T2 + BU*U2M3W2 + DW*W2M3U2 - SIGMAE
      FVEC(6) = CT*T2M3V2 - AV*V2M3T2 + DU*U2M3W2 - BW*W2M3U2 - SIGMAF
 999  RETURN
C
      END
```